DS 4400

Machine Learning and Data Mining I

Alina Oprea
Associate Professor, CCIS
Northeastern University

January 29 2019

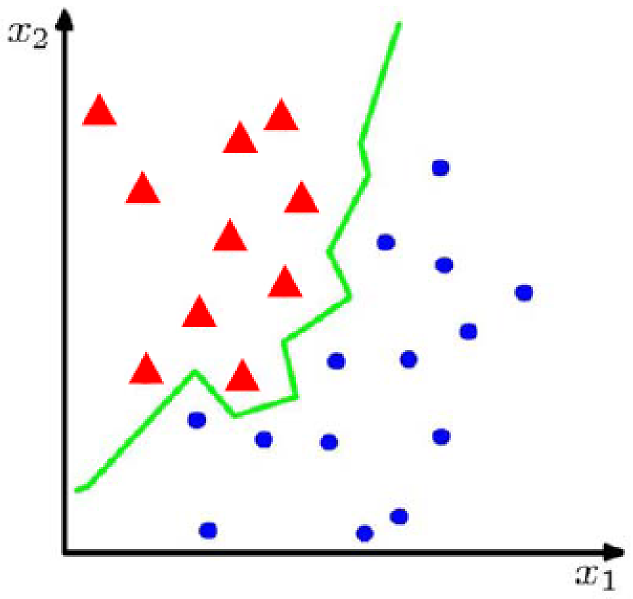
Review

- Gradient descent is an efficient algorithm for optimization and training LR
  - The most widely used algorithm in ML!
- More complex regression models exist
  - Polynomial, spline regression
- Regularization is general method to reduce model complexity and avoid overfitting
  - Add penalty to loss function
  - Ridge and Lasso regression

Outline

- Cross validation for parameter selection
- Linear Classification
- Perceptron
  - Online and batch perceptron
- Logistic regression
  - Classification based on probability
- Evaluation of classifiers
  - Metrics
  - ROC curves

Classification



Binary or discrete

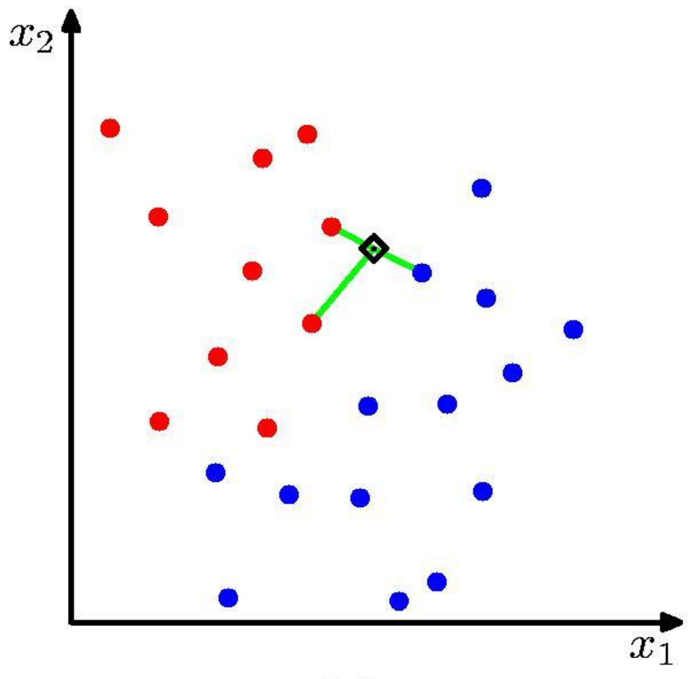
- Suppose we are given a training set of N observations

$$\{x^{(1)}, \dots, x^{(n)}\} \text{ and } \{y^{(1)}, \dots, y^{(n)}\}, x^{(i)} \in R^d, y^{(i)} \in \{-1, 1\}$$

- Classification problem is to estimate f(x) from this data such that

$$f\left(x^{(i)}\right) = y^{(i)}$$

kNN

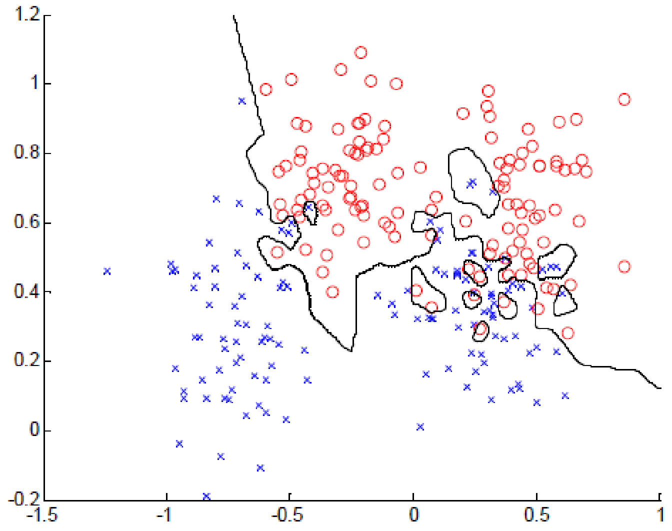


- Algorithm (to classify point x)
  - Find k nearest points to x (according to distance metric)
  - Perform majority voting to predict class of x
- Properties
  - Does not learn any model in training!
  - Instance learner (needs all data at testing time)

K = 1

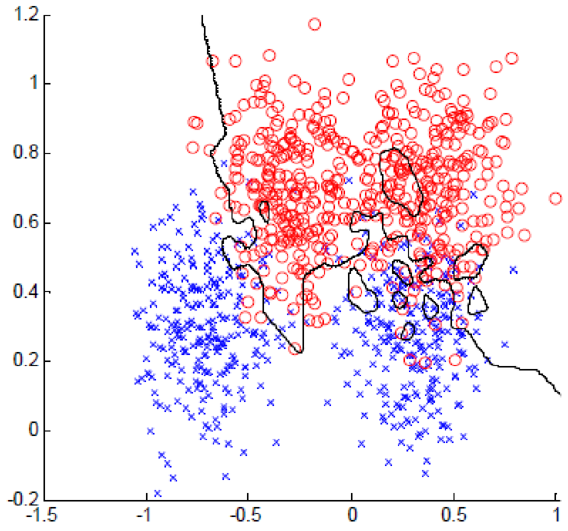
Overfitting!

Training data



error = 0.0

Testing data

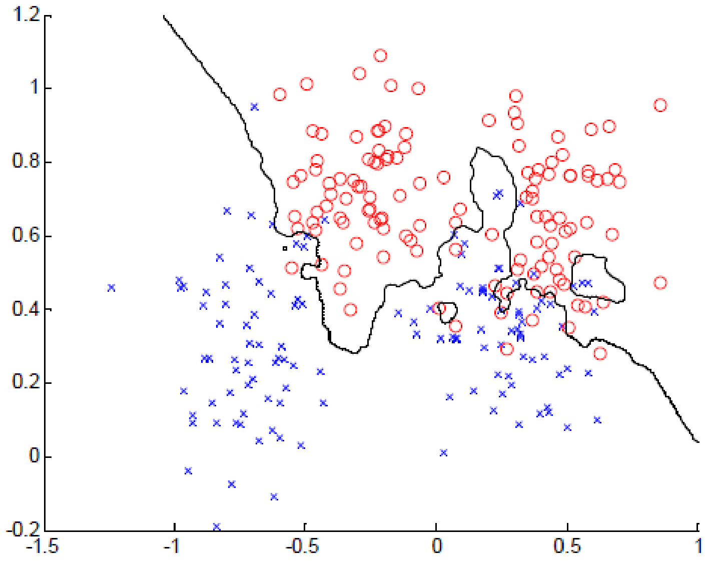


error = 0.15

How to choose k (hyper-parameter)?

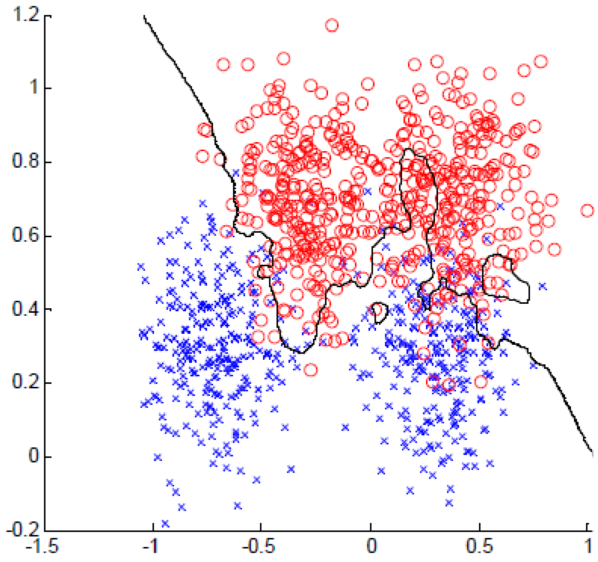
K = 3

Training data



error = 0.0760

Testing data

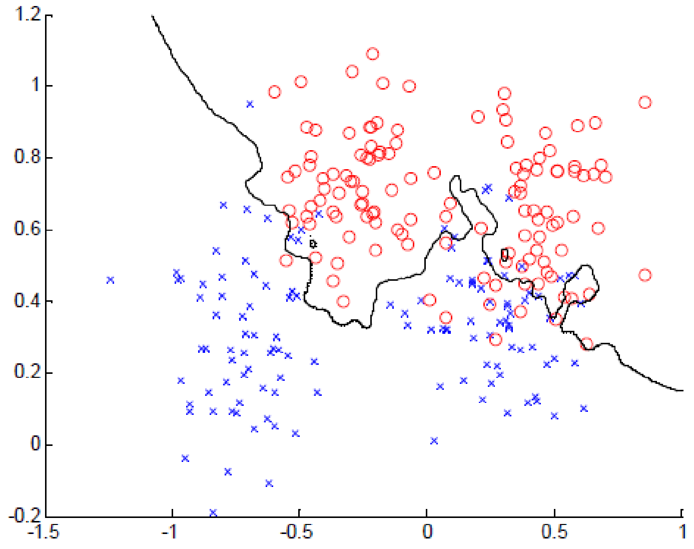


error = 0.1340

How to choose k (hyper-parameter)?

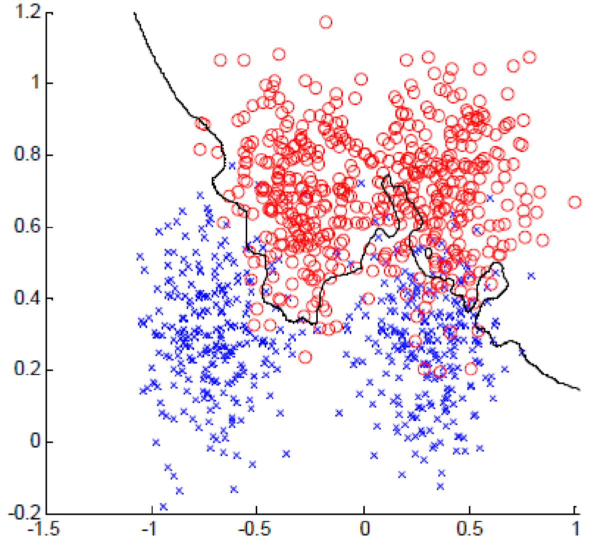
K = 7

Training data



error = 0.1320

Testing data



error = 0.1110

How to choose k (hyper-parameter)?

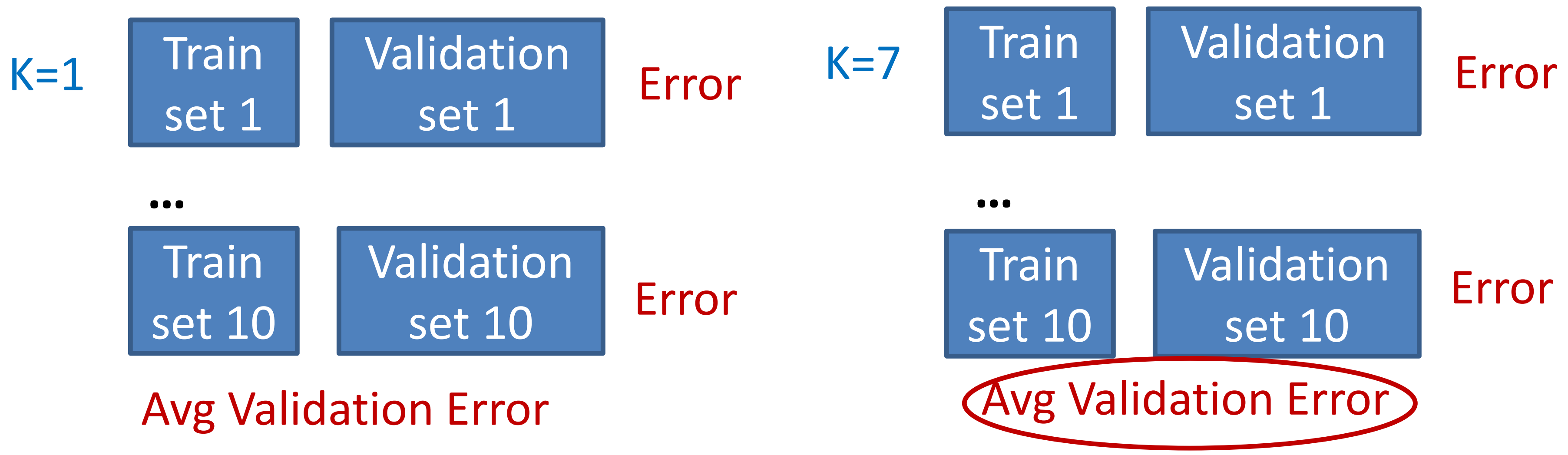
Cross-validation

As K increases:

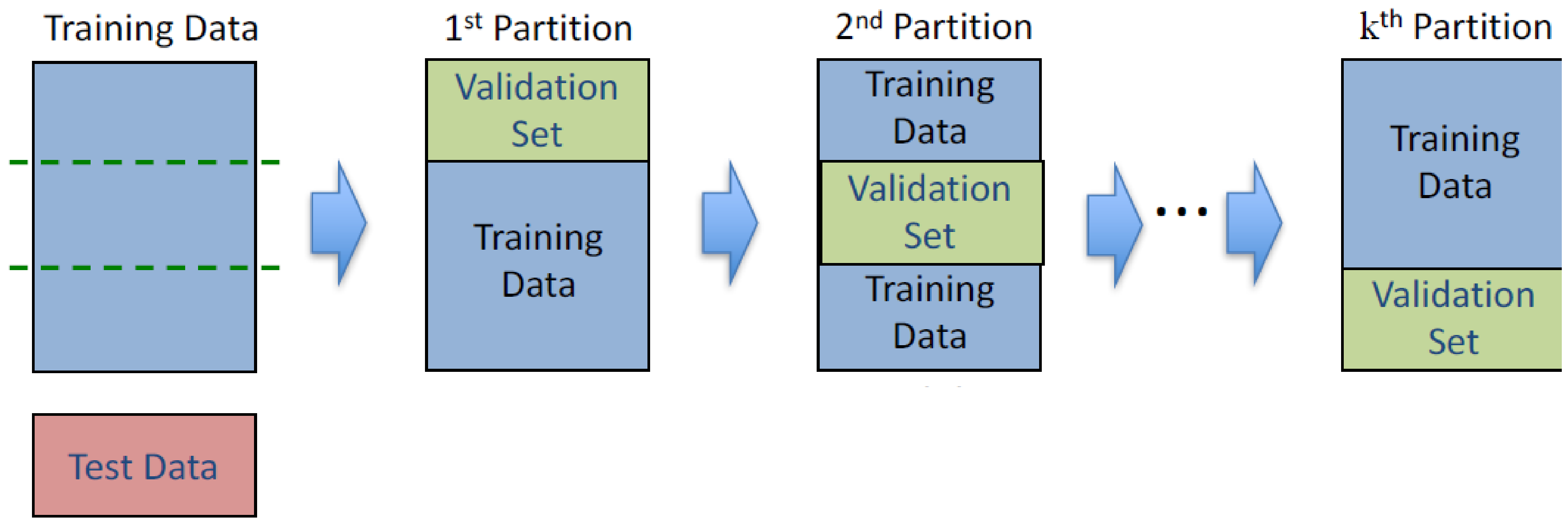
- Classification boundary becomes smoother
- Training error can increase

Choose (learn) K by cross-validation

- Split training data into training and validation
- Hold out validation data and measure error on this



Cross Validation



- k-fold CV
  - Split training data into k partitions (folds) of equal size
  - Pick the optimal value of hyper-parameter according to error metric averaged over all folds

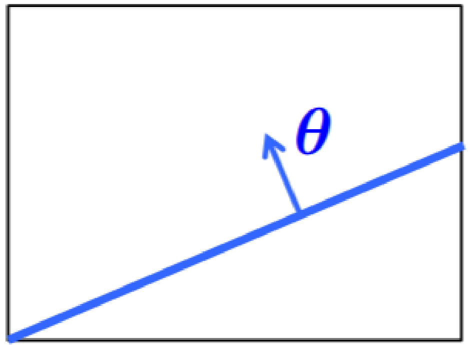
History of Perceptrons

- They were popularised by Frank Rosenblatt in the early 1960's.
  - They appeared to have a very powerful learning algorithm.
  - Lots of grand claims were made for what they could learn to do.
- In 1969, Minsky and Papert published a book called "Perceptrons" that analysed what they could do and showed their limitations.
  - Many people thought these limitations applied to all neural network models.
- The perceptron learning procedure is still widely used today for tasks with enormous feature vectors that contain many millions of features.

They are the basic building blocks for
Deep Neural Networks

Linear classifiers

- A **hyperplane** partitions space into 2 half-spaces
  - Defined by the normal vector $\boldsymbol{\theta} \in \mathbb{R}^{d+1}$
    - $\boldsymbol{\theta}$ is orthogonal to any vector lying on the hyperplane
  - Assumed to pass through the origin
    - This is because we incorporated bias term θ_0 into it by $x_0 = 1$

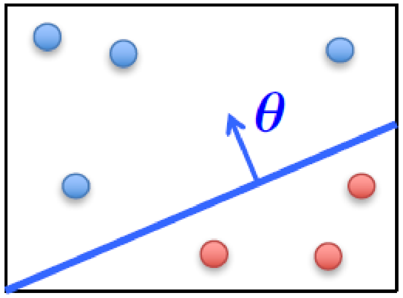


- Consider classification with +1, -1 labels ...

Linear classifiers

- **Linear classifiers**: represent decision boundary by hyperplane

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \quad \boldsymbol{x}^\mathsf{T} = \begin{bmatrix} 1 & x_1 & \ldots & x_d \end{bmatrix}$$



$$h(\boldsymbol{x}) = \text{sign}(\boldsymbol{\theta}^\mathsf{T}\boldsymbol{x}) \quad \text{where} \quad \text{sign}(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0 \end{cases}$$

– Note that: $\boldsymbol{\theta}^\mathsf{T}\boldsymbol{x} > 0 \implies y = +1$

$\boldsymbol{\theta}^\mathsf{T}\boldsymbol{x} < 0 \implies y = -1$

All the points x on the hyperplane satisfy: $\theta^T x = 0$

Example: Spam

- Imagine 3 features (spam is "positive" class):
  1. free (number of occurrences of "free")
  2. money (occurrences of "money")
  3. BIAS (intercept, always has value 1)

$$\sum_{i=0}^{d} x_i \theta_i$$

"free money"

x

```
BIAS  :  1
free  :  1
money :  1
...
```

θ

```
BIAS  : -3
free  :  4
money :  2
...
```

$$\begin{aligned}&(1)(-3) &+\\ &(1)(4) &+\\ &(1)(2) &+\\ &\ldots\\ &= 3\end{aligned}$$

$\sum_i \; x_i \theta_i > 0$ ➔ SPAM!!!

The Perceptron

$$h(\boldsymbol{x}) = \text{sign}(\boldsymbol{\theta}^\mathsf{T}\boldsymbol{x}) \quad \text{where} \quad \text{sign}(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0 \end{cases}$$

- The perceptron uses the following update rule each time it receives a new training instance $(\boldsymbol{x}^{(i)}, y^{(i)})$

$$\theta_j \leftarrow \theta_j - \frac{1}{2} \underbrace{\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)}_{\text{either 2 or -2}} x_j^{(i)}$$

  - If the prediction matches the label, make no change
  - Otherwise, adjust $\boldsymbol{\theta}$

The Perceptron

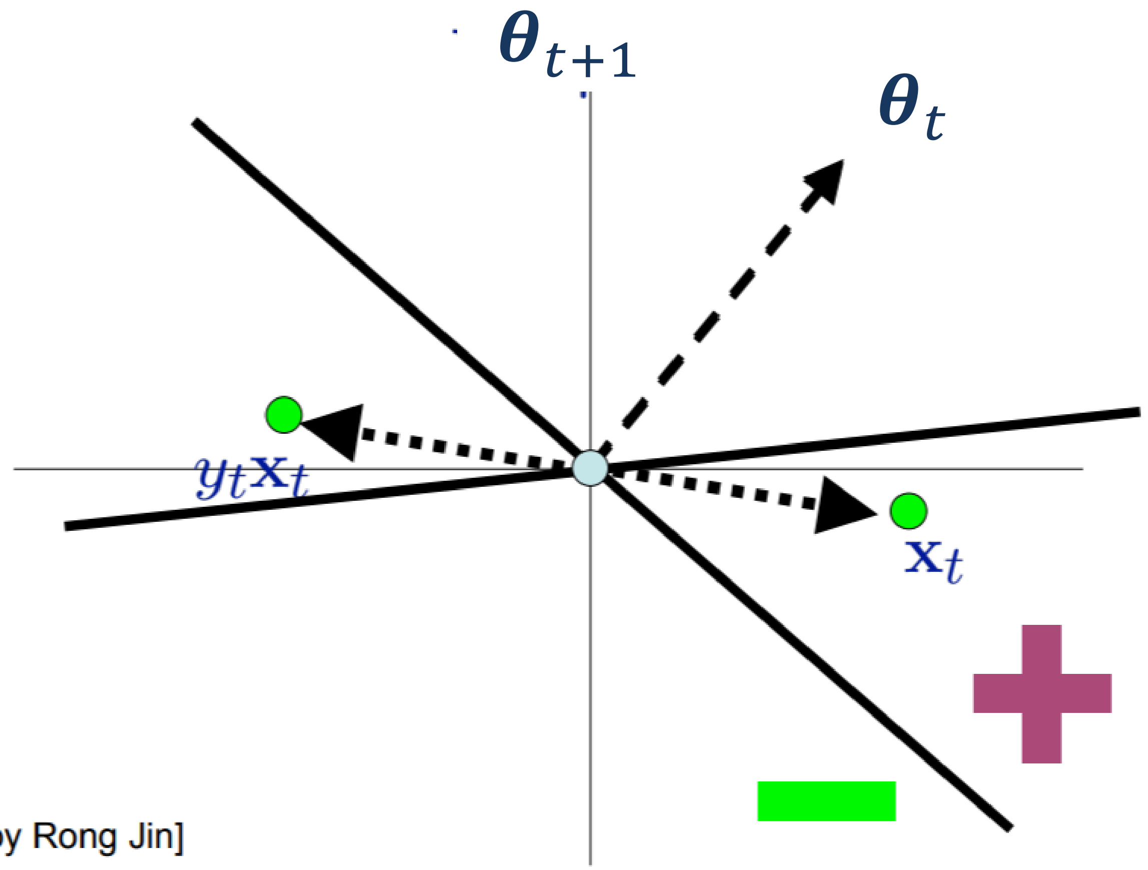
- The perceptron uses the following update rule each time it receives a new training instance $(\boldsymbol{x}^{(i)}, y^{(i)})$

$$\theta_j \leftarrow \theta_j - \frac{1}{2} \underbrace{\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)}_{\text{either 2 or -2}} x_j^{(i)}$$

- Re-write as $\theta_j \leftarrow \theta_j + \quad y^{(i)} x_j^{(i)}$ (only upon misclassification)

Perceptron Rule: If $\boldsymbol{x}^{(i)}$ is misclassified, do $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + y^{(i)} \boldsymbol{x}^{(i)}$

Geometric interpretation



[Slide by Rong Jin]

Online Perceptron

```
Let $\boldsymbol{\theta} \leftarrow [0, 0, \ldots, 0]$
Repeat:
    Receive training example $(\boldsymbol{x}^{(i)}, y^{(i)})$
    if $y^{(i)}\theta^T x^{(i)} \leq 0$                // prediction is incorrect
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + y^{(i)}\boldsymbol{x}^{(i)}$
```

Online learning – the learning mode where the model update is performed each time a single observation is received

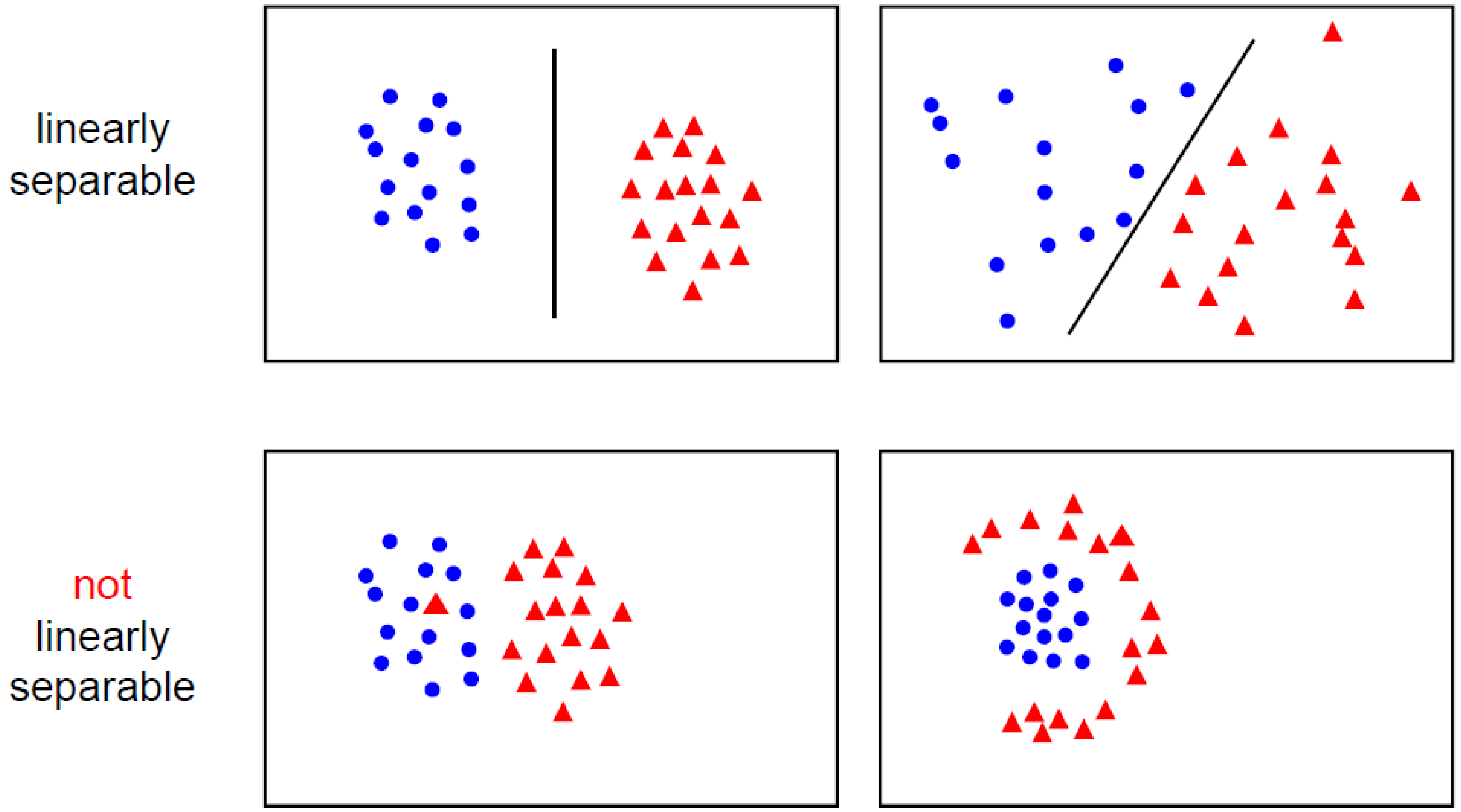
Batch learning – the learning mode where the model update is performed after observing the entire training set

Batch Perceptron

```
Given training data $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{n}$
Let $\boldsymbol{\theta} \leftarrow [0, 0, \ldots, 0]$
Repeat:
    Let $\boldsymbol{\Delta} \leftarrow [0, 0, \ldots, 0]$
    for $i = 1 \ldots n$, do
        if $y^{(i)} \theta^T x^{(i)} \leq 0$          // prediction for i$^{th}$ instance is incorrect
            $\boldsymbol{\Delta} \leftarrow \boldsymbol{\Delta} + y^{(i)} \boldsymbol{x}^{(i)}$
    $\boldsymbol{\Delta} \leftarrow \boldsymbol{\Delta}/n$                      // compute average update
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{\Delta}$
Until $\|\boldsymbol{\Delta}\|_2 < \epsilon$
```

Guaranteed to find separating hyperplane if data is linearly separable

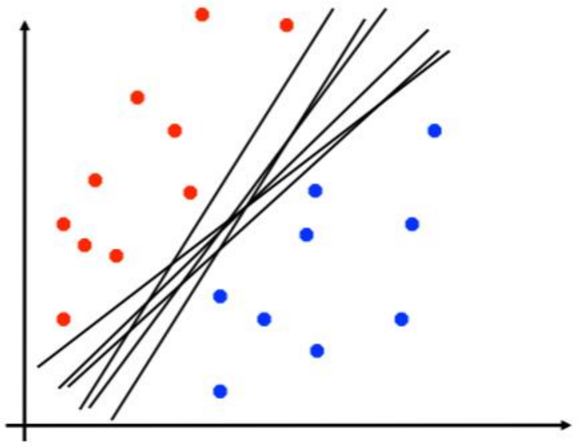
Linear separability



- For linearly separable data, can prove bounds on perceptron error (depends on how well separated the data is)

Perceptron Limitations

- Is dependent on starting point
- It could take many steps for convergence
- Perceptron can overfit
    - Move the decision boundary for every example



Which of this is optimal?

Improving the Perceptron

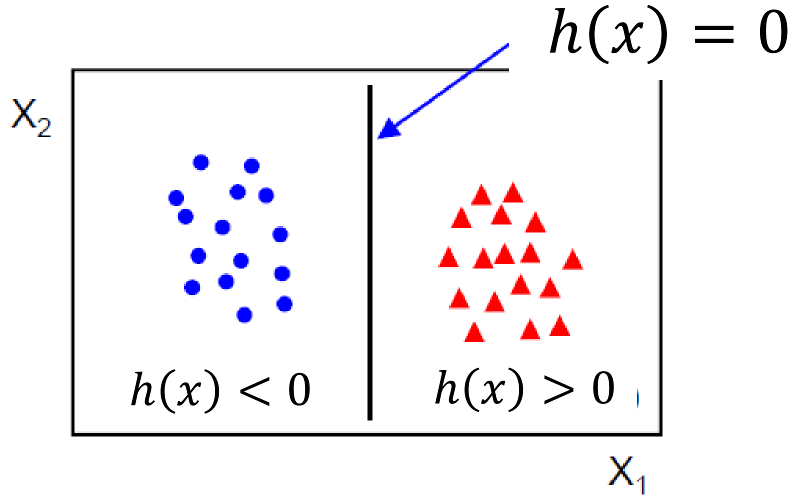
- The Perceptron produces many θ's during training
- The standard Perceptron simply uses the final θ at test time
  - This may sometimes not be a good idea!
  - Some other θ may be correct on 1,000 consecutive examples, but one mistake ruins it!

- **Idea:** Use a combination of multiple perceptrons
  - (i.e., neural networks!)
- **Idea:** Use the intermediate θ's
  - **Voted Perceptron**: vote on predictions of the intermediate θ's
  - **Averaged Perceptron**: average the intermediate θ's

Linear classifiers

A linear classifier has the form

$$h_\theta(x) = f(\theta^T x)$$



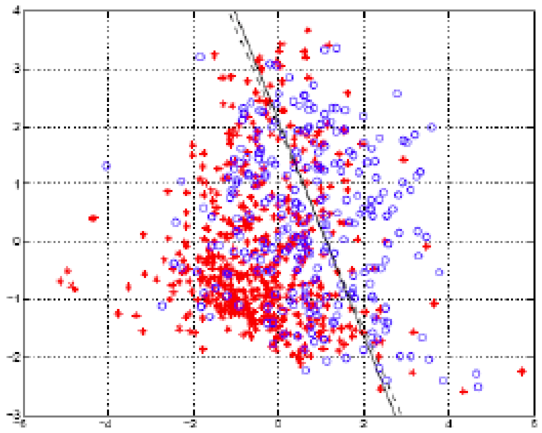
- Properties
  - $(\theta_0, \theta_1, \dots, \theta_d)$ = model parameters
  - Perceptron is a special case with $f = sign$
- Pros
  - Very compact model (size d)
  - Perceptron is fast
- Cons
  - Does not work for data that is not linearly separable

Classification based on Probability

- Instead of just predicting the class, give the probability of the instance being that class
  - i.e., learn $p(y \mid \boldsymbol{x})$
- Comparison to perceptron:
  - Perceptron doesn't produce probability estimate
- Recall that:

$$0 \leq p(\text{event}) \leq 1$$

$$p(\text{event}) + p(\neg\text{event}) = 1$$



Example

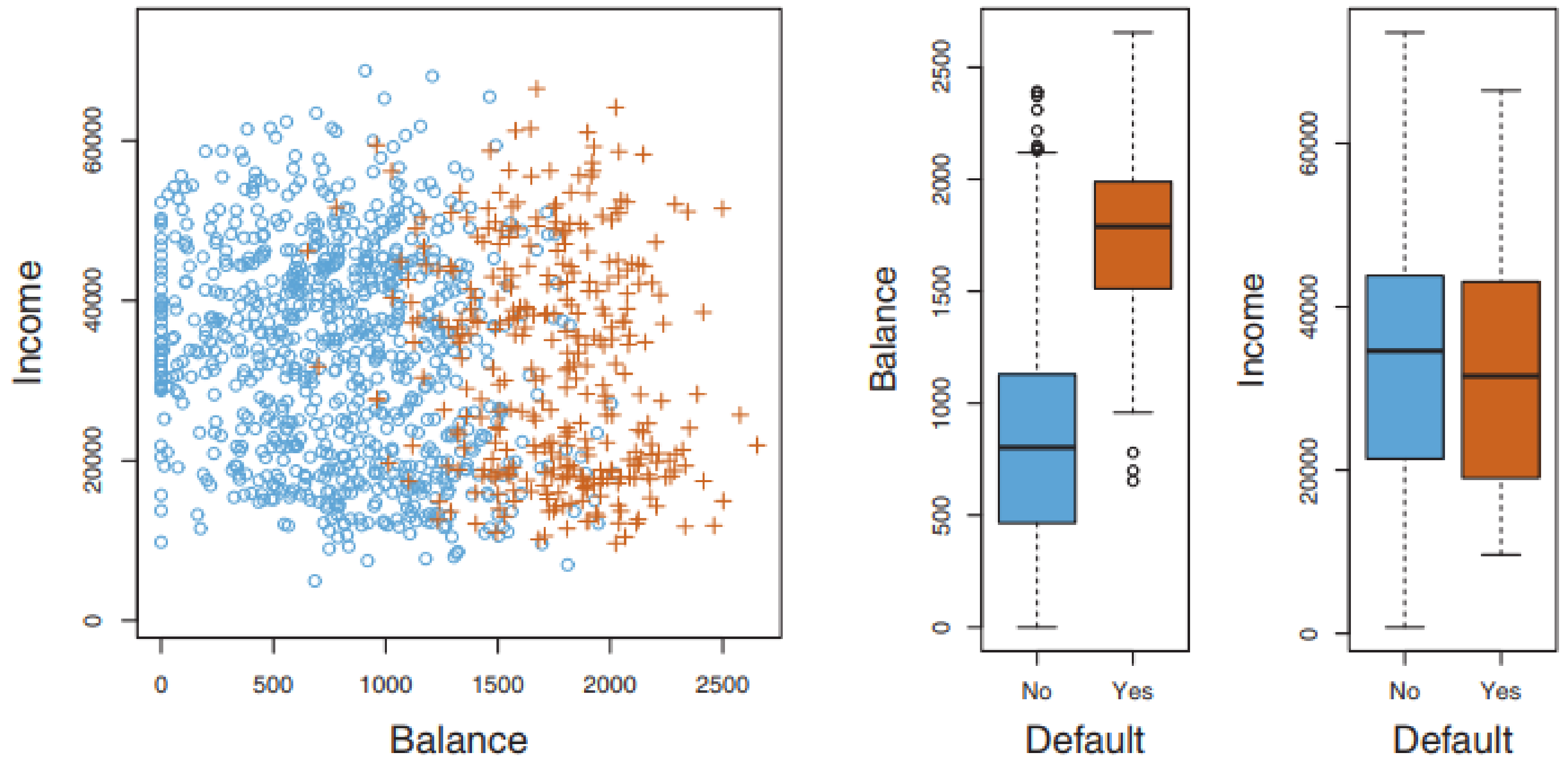


FIGURE 4.1. *The* Default *data set.* Left: *The annual incomes and monthly credit card balances of a number of individuals. The individuals who defaulted on their credit card payments are shown in orange, and those who did not are shown in blue.* Center: *Boxplots of* balance *as a function of* default *status.* Right: *Boxplots of* income *as a function of* default *status.*

Why not linear regression?

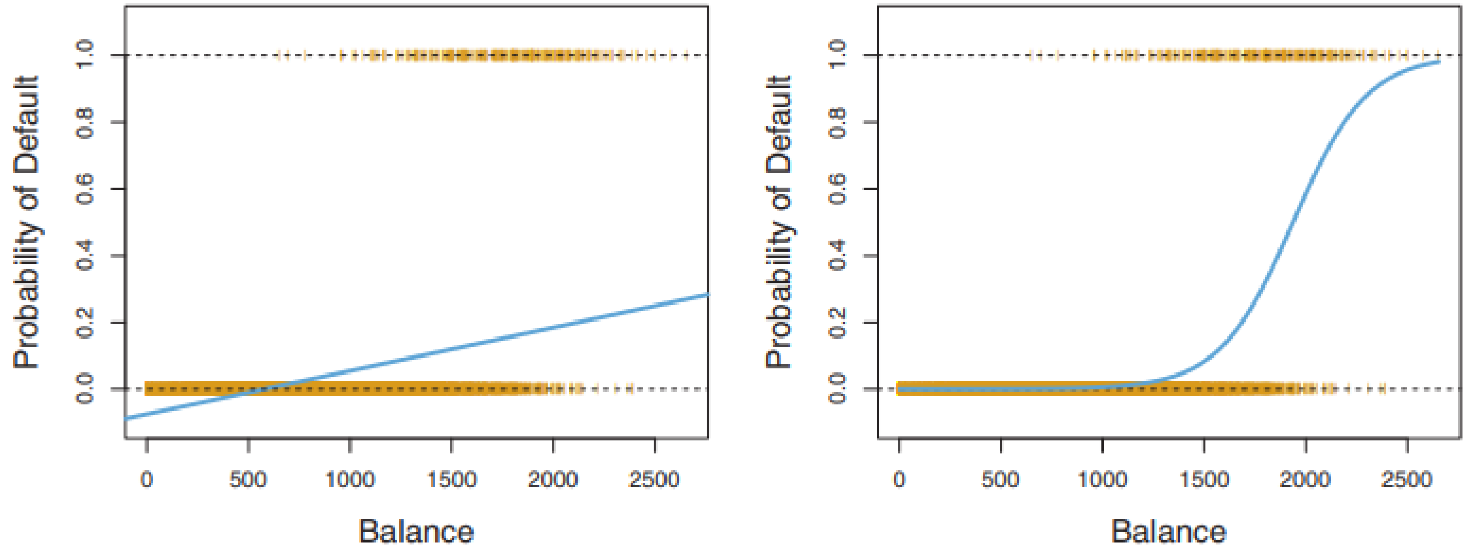


FIGURE 4.2. *Classification using the* `Default` *data.* Left: *Estimated probability of* `default` *using linear regression. Some estimated probabilities are negative! The orange ticks indicate the 0/1 values coded for* `default`*(*`No` *or* `Yes`*).* Right: *Predicted probabilities of* `default` *using logistic regression. All probabilities lie between* 0 *and* 1.

Logistic regression

- Takes a probabilistic approach to learning discriminative functions (i.e., a classifier)
- $h_{\boldsymbol{\theta}}(\boldsymbol{x})$ should give $p(y = 1 \mid \boldsymbol{x}; \boldsymbol{\theta})$
  - Want $0 \leq h_{\boldsymbol{\theta}}(\boldsymbol{x}) \leq 1$

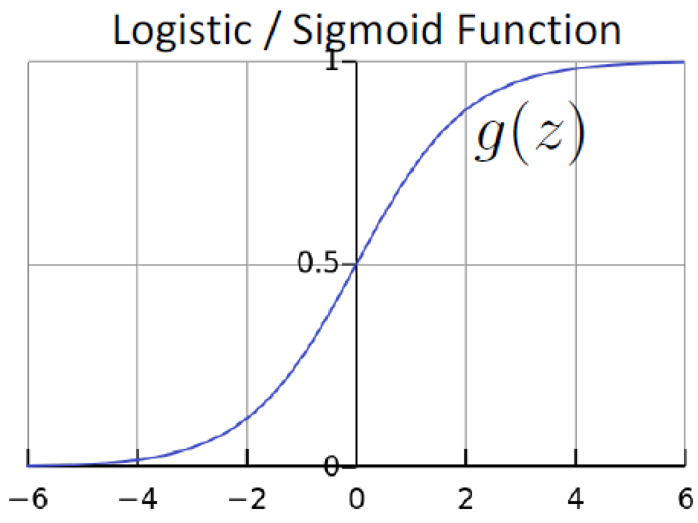
Can't just use linear regression with a threshold

- Logistic regression model:

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = g\left(\boldsymbol{\theta}^\intercal \boldsymbol{x}\right)$$

$$g(z) = \frac{1}{1 + e^{-z}}$$

$$\boxed{h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^\intercal \boldsymbol{x}}}}$$



Interpretation of Model Output

$h_{\boldsymbol{\theta}}(\boldsymbol{x})$ = estimated $p(y = 1 \mid \boldsymbol{x}; \boldsymbol{\theta})$

Example: Cancer diagnosis from tumor size

$$\boldsymbol{x} = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumorSize} \end{bmatrix}$$

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = 0.7$$

→ Tell patient that 70% chance of tumor being malignant

Note that: $p(y = 0 \mid \boldsymbol{x}; \boldsymbol{\theta}) + p(y = 1 \mid \boldsymbol{x}; \boldsymbol{\theta}) = 1$

Therefore, $p(y = 0 \mid \boldsymbol{x}; \boldsymbol{\theta}) = 1 - p(y = 1 \mid \boldsymbol{x}; \boldsymbol{\theta})$

Acknowledgements

- Slides made using resources from:
  - Andrew Ng
  - Eric Eaton
  - David Sontag
- Thanks!